Releasing and Patching ITOS

Integrated Test & Operations System
Release 6-x, $Date: 2006/03/21 16:07:25 $

Copyright 1999-2006, United States Government as represented by the Administrator of the National Aeronautics and Space Administration. No copyright is claimed in the United States under Title 17, U.S. Code.

This software and documentation are controlled exports and may only be released to U.S. Citizens and appropriate Permanent Residents in the United States. If you have any questions with respect to this constraint contact the GSFC center export administrator, <Thomas.R.Weisz@nasa.gov>.

This product contains software from the Integrated Test and Operations System (ITOS), a satellite ground data system developed at the Goddard Space Flight Center in Greenbelt MD. See <http://itos.gsfc.nasa.gov/> or e-mail <itos@itos.gsfc.nasa.gov> for additional information.

You may use this software for any purpose provided you agree to the following terms and conditions:
1. Redistributions of source code must retain the above copyright notice and this list of conditions.
2. Redistributions in binary form must reproduce the above copyright notice and this list of conditions in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product contains software from the Integrated Test and Operations System (ITOS), a satellite ground data system developed at the Goddard Space Flight Center in Greenbelt MD.

This software is provided ''as is'' without any warranty of any kind, either express, implied, or statutory, including, but not limited to, any warranty that the software will conform to specification, any implied warranties of merchantability, fitness for a particular purpose, and freedom from infringement and any warranty that the documentation will conform to their program or will be error free.

In no event shall NASA be liable for any damages, including, but not limited to, direct, indirect, special or consequential damages, arising out of, resulting from, or in any way connected with this software, whether or not based upon warranty, contract, tort, or otherwise, whether or not injury was sustained by persons or property or otherwise, and whether or not loss was sustained from or arose out of the results of, or use of, their software or services provided hereunder.

1 Releasing ITOS

Here are the steps in a release:

1. Make sure everyone knows a release is about to occur.
2. Make sure everything is checked in.
3. Update the release notes.
4. Tag the release.
5. Save the old '`/home/itos`'.
6. Export into '`/home/itos/src`'.
7. Build and test in '`/home/itos`'.
8. Apply last minute patches.
9. Package the release.
10. Install the release.
11. Configure the system.

1.1 Make sure everyone knows a release is about to occur

This is step is pretty simple – just tell the developers that a release is about to occur and that they shouldn't commit any more changes without your knowlege and consent.

The idea is that you don't want anything to change out from underneath you!

BTW, You might want to remind everyone that the release process may take several hours.

1.2 Make sure everything is checked in

When you're telling the developers the release is about to occur, ask them to make sure they've committed everything they intend to have released. To check for uncommitted changes:

In `csh` do

```
% cd /home/tcw/src
% cvs -n update |& grep ^M
```

or in `sh` do

```
$ cd /home/tcw/src
$ cvs -n update 2>&1 | grep ^M
```

The '`-n`' option tells `cvs` to only go through the motions and not to actually update anything; the '`grep ^M`' throws out all output except lines beginning with capital M – these lines identify files that are under CVS control and have been modified in '`/home/tcw`'.

You're looking for lines like

```
M lib/uppercase.c
M olstol/olstol.h
```

Each of these files has been modified in '`/home/tcw`' – you'll have to resolve each of these one way or another. The resolution will probably be to either delete the file or to commit the file.

1.3 Update the release notes

Finalize the release documentation in `src/doc/itos-relnotes.texi`.

1.4 Tag the release

Assuming the stuff in '`/home/itos`' looks OK, now's the time to tag the release.

To see the names of previous releases, do something like:

```
$ cd ~/tcw/src
$ cvs log Makefile | head -15

RCS file: /home/tcw/cvs/src/Makefile,v
Working file: Makefile
head: 2.6
branch:
locks: strict
access list:
symbolic names:
        Release_6-2: 2.6
        Build_6-1: 2.6
        trace-inst-tst: 2.5
        Demo: 2.5
        Release_6-0_alpha: 2.4
        start: 1.1.1.1
        tcw: 1.1.1
$
```

This shows that '`src/Makefile`' has previously been tagged '`Release_6-2`', '`Build_6-1`', '`trace-inst-tst`', '`Demo`', etc.

To tag the release do (assumes the tag '`Release_N-M`'):

```
$ cvs rtag Release_N-M src
```

It doesn't matter what directory you're in when you do this as long as your *CVSROOT* environment variable is set to '`cvsserverhost:/home/cvs/itos`':

```
$ echo $CVSROOT
cvsserverhost:/home/cvs/itos
```

where '`cvsserverhost`' is the development machine here in the ITOS Group. Tagging the release will take a while.

1.5 Save the old /home/itos

There's a chance – even though we've exhaustively tested everything – that the new release simply won't work. In that case, we'll need to revert back to the old system. So, before exporting the new release into '`/home/itos`', move the old '`/home/itos`' out of the way. This way, to restore the old system all you have to do is move it back.

As root on the development system, in '`/export/home`', run:

```
    # mv itos itos.6-3                  # move the old usr.tcw
    # mkdir itos                        # make the new usr.tcw
    # chown me:devel itos               # make it writeable by us
```

where you substitute the previous release number for '`6-3`', and your user name for '`me`'. If your umask is 022, also run "chmod 775 itos" so the group can write to it.

This proceedure is followed because '`/home/itos`' is an automounter mount point, so it's **not** OK to rename '`/home/itos`'. Instead, you've got to figure out the real name of the directory seen at '`/home/itos`'. Our '`/home/itos`' usually is mounted by the automounter from the development system's '`/export/home/itos`'.

1.6 Export into /home/itos/src

This assumes the new release is '`Release_6-4`':

```
    $ cd /home/itos
    $ cvs export -r Release_6-4 src
```

1.7 Build and test in /home/itos

Make sure that the correct Java is in your path (do '`java -version`') and that '`/usr/local/javalibs`' contains '`jcbwt220.jar`' and '`jcchartItos.jar`'. Then do:

```
    $ umask 002
    $ cd /home/itos/src
    $ ./bootstrap
    $ ./configure -prefix=/home/itos
    $ make install >install.log 2>&1
```

(The '`-prefix`' is really only needed on FreeBSD.)

Now run release test on the newly built release.

1.8 Apply last minute patches

Although it's never supposed to happen, occasionally this last little bit of testing uncovers a bug. You'd probably like to fix the bug and jam it into the release. Here's how:

1. Fix the bug. Keep track of all source file that changed.
2. Re-tag those source files. For example, if '`src/olstol/stProc.c`' had the last minute bug, you'd:

   ```
       $cvs rtag -F Release_6-3 src/olstol/stProc.c
   ```

   (The '`-F`' option tells CVS to overwrite the existing tag).
3. Re-export the source files into '`/home/itos/src`'. Continuing the above example:

   ```
       $ cd /home/itos
       $ rm src/olstol/stProc.c
       $ cvs export -r Release_6-3 src/olstol/stProc.c
   ```

4. Re-make whatever needs to be remade. The simplest way is to:

   ```
       $ cd /home/itos/src
       $ make install
   ```

1.9 Package the release

On the development machine, make the version file and the release tarfile and copy the tarfile to the distribution server:

```
$ cd /home/itos
$ echo "Release 6-2 for sparc-SunOS-5.6" >Version
$ mv src .src
$ tar zcf itos_6-2_sparc-SunOS-5.6.tar.gz *
$ mv .src src
$ scp itos_6-2_sparc-SunOS-5.6.tar.gz itos:~ftp/private/...
```

where '6-2' is the ITOS release number, 'sparc' is the processor type (`uname -p`), 'SunOS' is the operating system (`uname -s`), and '5.6' is the OS release number (`uname -r`) for which the ITOS was compiled. You can drop the 'RELEASE' from the FreeBSD release number.

Note that if we build the system specifically for the 64-bit UltraSPARC processor (that is, so it won't run on pre-V-9 SPARC chips), then replace the generic 'sparc' (`uname -p`) with the word 'ultrasparc'.

Now make the appropriate package:

1.9.1 Make the Source Tarball

On one of the development machines, make the source tarball:

```
$ cd /home/itos/src
$ make dist >dist.log 2>&1
$ mv itos-6.2.tar.gz ..
$ scp ../itos_6.2.tar.gz itos:~ftp...
```

1.9.2 Make a Solaris Package

On the Solaris development systems, make a 'pkginfo' file containing the following lines. Modify version information as appropriate.

```
PKG="ITOS"
NAME="Integrated Test & Operations System Release 6-2"
VERSION="6.2"
ARCH="sparc"
CATEGORY="utility"
BASEDIR="/home/itos"
DESC="ITOS is software for controlling and monitoring spacecraft or anything else that
VENDOR="NASA/GSFC"
EMAIL="itos@itos.gsfc.nasa.gov"
```

Now make a Solaris package of the release:

```
$ cd /home/itos
$ tar ztf itos_6-2_sparc-SunOS-5.6.tar.gz | pkgproto >proto
$ xemacs proto
$ pkgmk -o -r /home/itos -d . -f proto
```

When you edit the 'proto' file, add as the first line 'i pkginfo', and change all 644 modes to 664 and all 755 modes to 775. The result of all this is a directory called '/home/itos/ITOS'. Create another tarball of this and copy it to the FTP server:

```
$ tar zcf itos_6-2_sparc-SunOS-5.6-pkg.tgz ITOS
$ scp itos_6-2_sparc-SunOS-5.6-pkg.tgz itos:~ftp/pub/itos/Release_6-2
```

Use the '`.tgz`' extension like in FreeBSD to help further differentiate the package tarball from the from the other one.

1.9.3 Make a Red Hat Linux Package

On the Linux development system, in '`/home/itos`', make an '`rpmspec`' file containing the following, with the version number updated:

```
Summary: Spacecraft ground data system core
Name: ITOS
Version: 6.2
Release: 1
Copyright: Copyright 1999, 2000, 2001, 2002, United States Government
Group: Applications
Source: itos@itos.gsfc.nasa.gov
URL: http://itos.gsfc.nasa.gov/
Vendor: NASA/GSFC Code 584

%description
The Integrated Test and Operations System (ITOS) is software for
controlling and monitoring spacecraft or anything else that produces
telemetry and/or accepts commands.

It was developed at the Goddard Space Flight Center and, as of this
writing, is being used on 9 flight projects, including six of the
seven Small Explorers (SAMPEX, FAST, SWAS, TRACE, WIRE, HESSI), the
Ultra-Long Duration Balloon (ULDB) project, Triana, and Swift.  It
also was used on the shuttle-launched Spartan 201-05 mission carried
aboard STS-95.

%prep
%build
%install
%files
/home/itos/bin
/home/itos/classes
/home/itos/dbx
/home/itos/htdocs
/home/itos/lib
/home/itos/man
/home/itos/pages
/home/itos/procs
/home/itos/tcvol2
/home/itos/Version
/home/itos/Xdefaults
```

Now make an RPM package:

```
$ cd /home/itos
$ rpm -bb rpmspec
$ mv /usr/src/redhat/RPMS/i386/ITOS-7.1-1.rpm .
```

To install the package run

```
$ rpm -i -nodeps ITOS-7.1-1.rpm
```

1.9.4 Make a FreeBSD Package

On the FreeBSD development system, in '/home/itos', make a 'pkglist' file as follows:

```
$ cd /home/itos
$ rm /export/tmp/*
$ mv src itos_6-2_i386-FreeBSD-7.2.tar.gz /export/tmp
$ find . -type f -o -type l >pkglist
$ emacs pkglist
$ mv /export/tmp/* .
```

When editing the 'pkglist' file, remove the leading './' from each line, and add the line

```
@cwd /home/itos
```

as the first line in the file.

Create a 'comment' file with the following contents, updating the release number:

```
Integrated Test & Operations System (ITOS) Release 6-2
```

Create a 'descr' file with the following, updated as necessary:

```
The Integrated Test and Operations System (ITOS) is software for
controlling and monitoring spacecraft or anything else that produces
telemetry and/or accepts commands.

It was developed at the Goddard Space Flight Center and, as of this
writing, is being used on 9 flight projects, including six of the
seven Small Explorers (SAMPEX, FAST, SWAS, TRACE, WIRE, HESSI), the
Ultra-Long Duration Balloon (ULDB) project, Triana, and Swift.  It
also was used on the shuttle-launched Spartan 201-05 mission carried
aboard STS-95.
```

Finally, create the package:

```
$ pkg_create -c comment -d descr -f pkglist itos_6-2_i386-FreeBSD-4.4-pkg
```

The result is a file called 'itos_6-2_i386-FreeBSD-4.4-pkg.tgz', which is a package which can be installed with the pkg_add command.

1.10 Install the release

ITOS packages install in '/home/itos'. ITOS tarballs, obviously, can be installed anywhere. Before installing ITOS, be sure to remove the old installation or move it out of the way. If you are installing in an NFS mounted, but sure that directory is empty before performing the installation.

Choose from the following for instructions on installing ITOS on your platform:

Solaris package

```
tar zxpf itos_7-1_i386-FreeBSD-4.4-pkg.tgz
pkgadd -d . ITOS
```

Red Hat Linux package

```
rpm -i --nodeps ITOS-7.1-1.i386.rpm
```

FreeBSD package

```
pkg_add itos_7-1_i386-FreeBSD-4.4-pkg.tgz
```

Release tarball

```
cd /home/itos
tar zxpf itos_7-1_sparc-SunOS-5.7.tar.gz
```

Source tarball

```
tar zxpf itos-7.1.tar.gz
```

On ITOS build machines, the ITOS directory, '`/home/itos`', is mounted from '`/export/home/itos`' on the same machine. We need to install and test the package on each build machine to verify that the packaging process went OK. So, as root, do:

```
umount /home/itos
cd /export/home
mv itos itos_7-1_build
mkdir itos
chown user:group itos
chmod 775 itos
```

Now, while still root, install the package. Make sure no one is in the '`/home/itos`' directory or the `umount` will fail. Finally, re-run the release tests to validate the release package. When the release is validated, become root and restore the build directory as '`/home/itos`':

```
umount /home/itos
cd /export/home
mv itos itos_7-1_install
mv itos_7-1_build itos
```

1.11 Configure the system

1.11.1 Setting up the command relay

During spacecraft integration and test, the ITOS normally is deployed on a closed network. One ITOS workstation will have two ethernet adapters: one one the closed network and one on the open network. This workstation is used as a gateway through which spacecraft commands and telemetry data may be relayed to and from stations on the open network.

The telemetry relay is established by a STOL procedure called '`relay.proc`', which is distributed with the ITOS software.

The command relay is set up using `inetd`, so that the relay may be established by connecting to the correct port on the gateway workstation, assuming that the command destination port on the ITOS front end is available.

To set up the workstation for command relay, append the following to /etc/inetd.conf on the gateway workstation:

```
#
# ITOS command packet relay between open and closed network
cmdrelay stream tcp nowait nobody /usr/sbin/tcpd /home/itos/bin/relay scat 6000
```

Add the following line to the /etc/services file:

```
cmdrelay        9040/tcp                        # ITOS cmd relay server
```

Finally, make sure that '`scat`' appears in the hosts table as an alias for the actual SCAT computer hostname. Remember to run `make` in /var/yp on the NIS master server if you changed the hosts table.

Finally send a `SIGHUP` to the `inetd` process on the gateway computer. The command relay now should be available.

2 Patching ITOS

An ITOS patch consists of replacement programs, libraries, scripts, database inputs, documentation, and so on, along with the modified source files that went into building them. The source files are included primarily to facilitate debugging, and to allow the full system to be rebuilt.

Here are the steps for making an ITOS patch:

1. Switch to the '`-fixes`' source tree branch.
2. Document the patch.
3. List files for the patch.
4. Tag files involved in the patch.
5. Remove old files.
6. Export source files for the patch.
7. Build programs, libraries, etc.
8. Update the '`Version`' file.
9. Package the patch.
10. Re-create the patch for other architectures.
11. Test the patch for all architectures.
12. Install the patch on other systems.

2.1 Switch to the '`-fixes`' source tree branch.

We commit patches to a CVS branch to the source repository, never to the trunk (on which development is proceeding). Before you try this, read (at least skim) the CVS manual, chapter 5, Branches.

If, and **only** if, this is the first patch to a release, you need to create a branch to the source tree rooted at the release tag. To do this run '`cvs rtag -b -r Release_x-y Release_x-y_fixes src`', where '`x-y`' is the release number for which we are creating the patch.

Now you need to get the source from the branch. You can do this by cd'ing to a pristine work directory and running

```
$ cvs checkout -r Release_6-12_fixes src
```

(Of course, replace *6-12* with your actual release number!)

Now you can modify the source to fix the bug and commit the changes back to the branch.

2.2 Create the patch documentation

Add patch documentation to the release notes document '`src/doc/itos-relnotes.texi`'. Documentation for newer patches should be added *before* those for older ones.

Remember that you need to modify '`itos-relnotes.texi`' on the CVS branch for the release you're patching!

In some cases you will need to include instructions for a patch installer that go beyond the procedure given here. For example, if a patch requires that the database be rebuilt, you must include a comment to that effect. Put any such comments between the patch '@subheading' and the '@itemize' list of fixes.

```
@heading Patch 1
Feb 28, 1998

@subheading New Features and Bug Fixes

@itemize @bullet

@item The Sx interface (sxif) for the live_data_ingest (libldi) was not
      handling discrete conversions correctly.  This manifest in the
      configuration monitor (eqn_cfgmon).

@item Added security to the invocation daemon (invoked).  The program
      will execute tlmClient, frame_sorter, tlmPlay, and dsp_pktdump;
      and will kill only processes that it has started.

@item Removed obsolete call to dlopen() from wrapper library, and the
      "so" element from the Wrapper structure.

@item Fixed two bugs in end-of-session handling: In reassembler, moved
      frame vcid check to _after_ the check for EOS.  The previous
      configuration prevented EOS detection on VCs other than 0.  In
      sorter, added proper call to pool_adj_ref_count() to
      sorter_post_all().  Without this, some pools could have not been
      freed as they should be.

@end itemize
```

2.3 List files for the patch

In '/home/itos' on the main development machine, create a source manifest file containing a list of 'src' files involved in the patch. Names should be relative to the '/home/itos' directory. This list always will include the release notes file, since it contains the patch documentation. The file should be named 'patch??.src', where '??' is the 2-digit patch number. For example, the file 'patch01.src' might contain:

```
src/doc/itos-relnotes.texi
src/dsp/lib/sxif.c
src/util/invoked.c
src/lib/wrapper/wrap_anno12.c
src/lib/wrapper/wrap_ccsdstf.c
src/lib/wrapper/wrap_fep521.c
src/lib/wrapper/wrap_ftcp.c
src/lib/wrapper/wrap_itp.c
src/lib/wrapper/wrap_smex.c
```

```
src/lib/wrapper/wrapper.c
src/lib/wrapper/wrapper.h
src/tm/pkt/Makefile
src/tm/pkt/frame_input.c
src/tm/pkt/frame_sorter.c
src/tm/pkt/frame_sorter.h
src/tm/pkt/reassembler.c
src/tm/pkt/sorter.c
```

In the same directory, also create a manifest containing the list all target files that will make up the patch – executables, libraries, scripts, etc. Name it the same as the source manifest, but without the '`.src`' extension. For example, '`patch01`' is created as a companion to '`patch01.src`' and looks like this:

```
Version
patch01
htdocs/itos-relnotes
htdocs/itos-relnotes.ps
lib/libldi.a
lib/libldi.so
bin/invoked
lib/libtmw.a
lib/libtmw.so
bin/frame_sorter
```

Our convention is to begin with the '`Version`' file, followed by the manifest, and then each target file, beginning the release notes targets. (Note that '`htdocs/itos-relnotes`' is a directory.)

2.4 Tag files involved in the patch

If multiple developers are contributing to the patch, stop here until everyone reaches this step. Then one developer should complete the patch procedure from here.

Use the source manifest file to tag the source files in the patch. From '`/home/itos`' on the development machine, run

```
xargs cvs rtag -r Release_6-7_fixes R_6-7_Patch_01 < patch01.src
```

to tag source files for Patch_01 to Release_6-7, for example.

Notice the tag name convention: The form of the original tag for any release is '`Release_x-y`'. For each patch to that release, the form is '`R_x-y_Patch_z`', where '`x-y`' is the release version number, and '`z`' is the patch number.

2.5 Remove old files

Given the source manifest '`patch01.src`' containing a list of sources involved in the patch, run on the development machine in '`/home/itos`'

```
xargs rm <patch01.src
```

to remove the old source files. This is necessary since '`cvs`' will not export over a read-only file, which include all source files in the source tree.

2.6 Export source files for the patch

From /home/itos on the development machine, export the patch source files. The example is for Patch_01 to Release_6-7.

```
xargs cvs export -r R_6-7_Patch_01 <patch01.src
```

2.7 Build programs, libraries, etc

Move to /home/itos/src on the development machine and run '`make install >patch01.log 2>&1`'. This will build any programs, libraries, and so on affected by the patch and install them in /home/itos/bin, /home/itos/lib, etc. Redirect the output into a log file, as shown here for a Bourne-style shell.

2.8 Update the '`Version`' file

Modify the file '`/home/itos/Version`', so that it reflects the patch we're making. The recommended form for the contents of the '`Version`' file is '`Release x-y patchlevel z for sparc-SunOS-5.6`', where '`x-y`' is the release version number, and '`z`' is the number of the new patch. The trailing text, '`sparc-SunOS-5.6`', gives the processor and operating system for which the release was built, and will vary accordingly.

2.9 Package the patch

In /home/itos on the development machine, run something like

```
tar zcfT Patch_01_sparc-SunOS-5.6.tar.gz patch01
```

to create the patch file, '`Patch_01_sparc-SunOS-5.6.tar.gz`'. The '`patch01`' is the tarfile manifest we create earlier. The patch tarfile name is formed in a manner similar to that used to name the release tarfile.

Also, only on the first system on which a particular patch is being created, create a source patch. From '`/home/itos`', run:

```
tar zcfT Patch_01_src.tar.gz patch01.src
```

replacing the patch number, as appropriate.

Copy the patch file(s) to '`/home/ftp/pub/itos/Release_`*vers*' on the ftp server, where *vers* is the ITOS version number for which you've created the patch. If the directory doesn't exist, create it.

2.10 Re-create the patch for other architectures

We also need to re-create the patch on each architcture we support. Begin the process by copying the manifest files from the patch to a development machine of the desired architecture.

Note that on FreeBSD systems, shared libraries are required to have version numbers, so you will have to add the appropriate version numbers to the names of any shared libraries in the patch tarfile manifest.

Now on the other build machines go to step 5 and follow the procedure back to this point to re-create the patch for the target architecture.

2.11 Test the patch for all architectures

Test the patch on each supported architecture. If the patch is large, or touches on key components, run the complete ITOS release test procedure.

If isolated problems are discovered during testing, and the patch needs to be modified, commit the modifications to the fixes branch, and make the patch tag point to the new revisions. To re-tag the patch do this:

```
xargs cvs rtag -F -r Release_6-7_fixes R_6-7_Patch_01 < patch01.src
```

The patch tag will now point to the latest revisions on the patch branch. Then go to step 5 of the patch procedure to re-build and re-test the patch.

2.12 Install the patch on other systems

Patches must be installed in numerical order, and patches must not be skipped. You must, for example, install patch 3 after patch 2, and you must install patches 1 and 2 before installing patch 3.

On any test conductor workstation, `ftp` to the ftp server and retrieve the patch file into '`/home/itos`'. Then unpack the file with the command '`tar zxpmf Patch_`*xx*`.tar.gz`', where *xx* is the patch number.

The `tar` option '`p`' preserves the permissions and the option '`m`' updates the modification time.

Finish up by performing any steps listed in the '`Special Instructions`' section of the patch document, if any.

The patch now is installed.

Table of Contents